IN THE CLAIMS

Please amend the claims as follows:

Claim 1 (Currently Amended): A system for program language processing for translating source programs to generate an object program, comprising:

a preprocessor configured to execute preprocessing of source programs inputted in translation units;

a data type definition table, arranged for one object program, configured to store a set of a name of data type definition for data or a function in the source program ~~programs,~~ and a use flag of the name ~~set in a use status when the corresponding data type definition is described in a body of any of all source programs to be linked to the one object program~~;

a data type definition detector configured to detect a predetermined data type definition declared in the preprocessed source program;

a first table updating module configured to, if a name of the detected data type definition is not registered, register the name of the detected data type definition into the data type definition table;

a first source updating module configured to, if the name of the detected data type definition is registered, delete the data type definition in the source program;

a second table updating module configured to, if the data type definition is described in a body of any of the source programs to be linked into the one object program, set the use flag to a use status;

a second source updating module configure to delete the data type definition of which the use flag is set to the use status from all the source programs to optimize the source programs;

~~a code optimizing processor configured to scan all the preprocessed source programs to be used as a source for generating the object program, and delete, when a data type~~

~~definition is already stored or the use flag is not set in a use status in the data type definition table, the data type definition from the source programs to optimize the source programs;~~

a language processor configured to compile the optimized source programs; and

a software driver configured to control a transfer of a source program and a processing result of at least one of the preprocessor, the code optimizing processor, and the language processor.

Claim 2 (Canceled).

Claim 3 (Currently Amended): The system according to claim 1 [[2]], ~~wherein the code optimizing processor~~ further comprising ~~includes~~:

an instantiation request detector configured to detect an instantiation request of a data type definition in ~~from~~ the preprocessed source program;

a third table updating module configured to, if instantiation information arranged for each data type of multiphase data type is not registered, register the instantiation information into the data type definition table, the multiphase data type employing a template model for a various data types to be instantiated; and

a third source updating module configured not to, if the instantiation information of the data type of which instantiation is requested is not registered, generate the instance of the data type definition in the source program.

~~a second decision module configured to decide, with reference to instantiation information in the data type definition table, whether the instance of a data type definition corresponding to the detected instantiation request of the data type definition has been generated or not; and~~

~~an instance generator configured to generate the instance of the data type definition when it is decided by the second decision module that the instance of the data type definition has not been generated, register information representing the generation of the instance into the data type definition table as the instantiation information, and~~

~~not generate the instance of the data type definition when it is decided by the second decision module that the instance of the data type definition has been generated.~~

Claims 4-6 (Canceled).

Claim 7 (Currently Amended): The system according to claim 3, [[6,]] wherein the data type definition table includes member usage information representing, when the data type is the multiphase type holding member function, whether each member function is used or not, and

the third source updating module ~~optimizing processor~~ determines member function of the multiphase type the instance of which is to be actually generated in the source program with reference to the member usage information in the data type definition table.

Claim 8 (Currently Amended): The system according to claim 3, wherein the third source updating module ~~instance generator of the code optimizing processor~~ converts the name of the data definition into an unique name in a source program.

Claim 9 (Currently Amended): A method of program language processing for translating source programs to generate an object program, comprising:

executing preprocessing of source program inputted in translation units;

generating a data type definition table, arranged for one object program, for storing a set of a name of data type definition for data or a function in the source program ~~programs,~~ and a use flag of the name ~~set in a use status when the corresponding data type definition is described in a body of any of all source programs to be linked to the one object program~~;

detecting a predetermined data type definition declared in the preprocessed source program;

registering, if a name of the detected data type definition is not registered, the name of the detected data type definition into the data type definition table;

deleting, if the name of the detected data type definition is registered, the data type definition in the source program;

setting, if the data type definition is described in a body of any of the source programs to be linked into the one object program, the use flag to a use status;

deleting the data type definition of which the use flag is set to the use status from all the source programs to optimize the source programs;

~~scanning all the preprocessed source programs to be used as a source for generating the object program;~~

~~deleting, when a data type definition is already stored or the use flag is not set in a use status in the data type definition table, the data type definition from the source program to optimize the source program;~~ and

compiling the optimized source program.

Claim 10 (Canceled).

Claim 11 (Currently Amended): The method according to claim ~~10~~ 9, ~~wherein the code optimizing step~~ further comprising ~~includes~~:

detecting an instantiation request of a data type definition in ~~from~~ the preprocessed source program;

registering, if instantiation information arranged for each data type of a multiphase data type is not registered, the instantiation information into the data type definition table, the multiphase data type employing a template model for various data types to be instantiated; and

generating, if the instantiation information of the data type of which instantiation is requested is not registered, no instance of the data type definition in the source program.

~~deciding, with reference to instantiation information in the data type definition table, whether the instance of a data type definition corresponding to the detected instantiation request of the data type definition has been generated or not; and~~

~~generating the instance of the data type definition when it is decided that the instance of the data type definition has not been generated, registering information representing the generation of the instance into the data type definition table as the instantiation information, and~~

~~not generating the instance of the data type definition when it is decided that the instance of the data type definition has been generated.~~

Claims 12-14 (Canceled).

Claim 15 (Currently Amended): The method according to claim 11, [[14,]] wherein the data type definition table includes member usage information representing, when the data type is the multiphase type holding member function, whether each member function is used or not, and

the generating ~~optimizing~~ step determines member functions of the multiphase type the instance of which must be actually generated in the source program with reference to the member usage information in the data type definition table.

Claim 16 (Currently Amended): A computer readable recording medium for causing a computer to execute program language processing for translating a source program to generate an object program, comprising:

a process for executing preprocessing of source program input in translation units;

a process for generating a data type definition table, arranged for one object program, for storing a set of a name of data type definition for data or a function in the source program ~~programs,~~ and a use flag of the name ~~set in a use status when the corresponding data type definition is described in a body of any of all source programs to be linked to the one object program~~;

a process for detecting a predetermined data type definition declared in the preprocessed source program;

a process for registering, if a name of the detected data type definition is not registered, the name of the detected data type definition into the data type definition table;

a process for deleting, if the name of the detected data type definition is registered, the data type definition in the source program;

a process for setting, if the data type definition is described in a body of any of the source programs to be linked into the one object program, the use flag to a use status;

a process for deleting the data type definition of which the use flag is set to the use status from all the source programs to optimize the source programs;

~~a process for scanning all the preprocessed source programs to be used as a source for generating the object program;~~

~~a process for deleting, when a data type definition is already stored or the use flag is not set in a use status in the data type definition table, the data type definition from the source programs with reference to a data type definition which has been instantiated as needed to optimize the source program;~~ and

a process for compiling the optimized source program.

Claim 17 (Canceled).

Claim 18 (Currently Amended): The medium according to claim ~~17~~ 16, ~~wherein the code optimizing process~~ further comprising ~~includes~~:

a process for detecting an instantiation request of a data type definition in ~~from~~ the preprocessed source program;

a process for registering, if instantiation information arranged for each data type of a multiphase data type is not registered, the instantiation information into the data type definition table, the multiphase data type employing a template model for various data types to be instantiated; and

a process for generating, if the instantiation information of the data type of which instantiation is requested is not registered, no instance of the data type definition in the source program.

~~a process for deciding, with reference to instantiation information in the data type definition table, whether the instance of a data type definition corresponding to the detected instantiation request of the data type definition has been generated or not; and~~

~~a process for generating the instance of the data type definition when it is decided that the instance of the data type definition has not been generated, registering information~~

~~representing the generation of the instance into the data type definition table as the instantiation information, and~~

~~not generating the instance of the data type definition when it is decided that the instance of the data type definition has been generated~~.

Claim 19 (Currently Amended): A program product for causing a computer to execute program language processing for translating source programs to generate an object program, comprising:

a process for executing preprocessing of source ~~program~~ programs input in translation units;

a process for generating a data type definition table, arranged for one object program, for storing a set of a name of data type definition for data or a function in the source program ~~programs,~~ and a use flag of the name ~~set in a use status when the corresponding data type definition is described in a body of any of all source programs to be linked to the one object program~~;

a process for detecting a predetermined data type definition declared in the preprocessed source program;

a process for registering, if a name of the detected data type definition is not registered, the name of the detected data type definition into the data type definition table;

a process for deleting, if the name of the detected data type definition is registered, the data type definition in the source program;

a process for setting, if the data type definition is described in a body of any of the source programs to be linked into the one object program, the use flag to a use status;

a process for deleting the data type definition of which the use flag is set to the use status from all the source programs to optimize the source programs;

~~a process for scanning all the preprocessed source programs to be used as a source for generating the object program;~~

~~a process for deleting, when a data type definition is already stored or the use flag is not set in a use status in the data type definition table, the data type definition from the source programs to optimize the source program;~~ and

a process for compiling the optimized source program in units of translation.

Claim 20 (Canceled).

Claim 21 (Currently Amended): A system for program language processing for translating source programs to generate an object program, comprising:

a preprocessor for executing preprocessing of source programs inputted in translation units;

a multiphase data type definition table, arranged for one object program, for storing a set of a name of multiphase data type definition ~~employing a template model~~ for data or a function in the source program ~~programs,~~ and a use flag of the name ~~set in a use status when the corresponding data type definition is described in a body of any of all source programs to be linked to the one object program~~, the multiphase data type employing a template model for various data types to be instantiated;

a data type definition detector configured to detect the multiphase data type definition declared in the preprocessed source program;

a first table updating module configured to, if a name of the detected data type definition is not registered, register the name of the detected data type definition into the multiphase data type definition table;

a first source updating module configured to, if the name of the detected data type definition is registered, delete the multiphase data type definition in the source program;

a second table updating module configured to, if the multiphase data type definition is described in a body of any of the source programs to be linked into the one object program, set the use flag to a use status;

a second source updating module configure to delete the multiphase data type definition of which the use flag is set to the use status from all the source programs to optimize the source programs;

a third table updating module configured to, if instantiation information arranged for each data type of a multiphase data type is not registered, register the instantiation information into the data type definition table, the multiphase data type employing a template model for various data types to be instantiated;

a third source updating module configured not to, if the instantiation information of the data type of which instantiation is requested is not registered, generate the instance of the data type definition in the source program;

~~a first code optimizing processor for scanning all the preprocessed source programs to be used as a source for generating the object program, and deleting, when a multiphase data type definition is already stored or the use flag is not set in a use status in the multiphase data type definition table, the multiphase data type definition from the source programs to optimize the source programs;~~

~~a second code optimizing processor for scanning all the preprocessed source programs, and generating instance of the multiphase data type description in the body of the source programs only when the corresponding use flag is not set in a use status in the multiphase data type definition table;~~

a language processor for compiling the optimized source programs; and

a software driver for controlling a transfer of a source program and a processing result of at least one of the preprocessor, the code optimizing processor, and the language processor.

Claim 22 (Currently Amended): A method of program language processing for translating source programs to generate an object program, comprising:

executing preprocessing of source programs inputted in translation units;

generating a multiphase data type definition table, arranged for one object program, for storing a set of a name of multiphase data type definition ~~employing a template model~~ for data or a function in the source program ~~programs,~~ and a use flag of the name ~~set in a use status when the corresponding data type definition is described in a body of any of all source programs to be linked to the one object program~~, the multiphase data type employing a template model for various data types to be instantiated;

detecting the multiphase data type definition declared in the preprocessed source program;

registering, if a name of the detected data type definition is not registered, the name of the detected data type definition into the multiphase data type definition table;

deleting, if the name of the detected data type definition is registered, the multiphase data type definition in the source program;

setting, if the multiphase data type definition is described in a body of any of the source programs to be linked into the one object program, the use flag to a use status;

deleting the multiphase data type definition of which the use flag is set to the use status from all the source programs to optimize the source programs;

registering, if instantiation information arranged for each data type of a multiphase data type is not registered, the instantiation information into the data type definition table, the multiphase data type employing a template model for various data types to be instantiated;

generating, if the instantiation information of the data type of which instantiation is requested is not registered, no instance of the data type definition in the source program;

~~scanning all the preprocessed source programs to be used as a source for generating the object program;~~

~~deleting, when a multiphase data type definition is already stored or the use flag is not set in a use status in the multiphase data type definition table, the multiphase data type definition from the source programs to optimize the source programs;~~

~~generating instance of the multiphase data type description in the body of the source programs only when the corresponding use flag is not set in a use status in the multiphase data type definition table;~~ and

compiling the optimized source programs.

Claim 23 (Currently Amended): A computer readable recording medium for causing a computer to execute program language processing for translating source ~~program~~ programs to generate an object program, comprising:

a process for executing preprocessing of source programs inputted in translation units;

a process for generating a multiphase data type definition table, arranged for one object program, for storing a set of a name of multiphase data type definition ~~employing a template model~~ for data or a function in the source program ~~programs,~~ and a use flag of the name ~~set in a use status when the corresponding data type definition is described in a body of any of all source programs to be linked to the one object program~~, the multiphase data type employing a template model for various data types to be instantiated;

a process for detecting the multiphase data type definition declared in the preprocessed source program;

a process for registering, if a name of the detected data type definition is not registered, the name of the detected data type definition into the multiphase data type definition table;

a process for deleting, if the name of the detected data type definition is registered, the multiphase data type definition in the source program;

a process for setting, if the multiphase data type definition is described in a body of any of the source programs to be linked into the one object program, the use flag to a use status;

deleting the multiphase data type definition of which the use flag is set to the use status from all the source programs to optimize the source programs;

registering, if instantiation information arranged for each data type of a multiphase data type is not registered, the instantiation information into the data type definition table, the multiphase data type employing a template model for various data types to be instantiated;

generating, if the instantiation information of the data type of which instantiation is requested is not registered, no instance of the data type definition in the source program;

~~a process for scanning all the preprocessed source programs to be used as a source for generating the object program;~~

~~a process for deleting, when a multiphase data type definition is already stored or the use flag is not set in a use status in the multiphase data type definition table, the multiphase data type definition from the source programs to optimize the source programs;~~

~~a process for generating instance of the multiphase data type description in the source programs only when the corresponding use flag is not set in a use status in the multiphase data type definition table;~~ and

a process for compiling the optimized source programs.

Claim 24 (Currently Amended): A program product for causing a computer to execute program language processing for translating source programs to generate an object program, comprising:

a process for executing preprocessing of source programs inputted in translation units;

a process for generating a multiphase data type definition table, arranged for one object program, for storing a set of a name of multiphase data type definition ~~employing a template model~~ for data or a function in the source program ~~programs,~~ and a use flag of the name ~~set in a use status when the corresponding data type definition is described in a body of any of all source programs to be linked to the one object program~~, the multiphase data type employing a template model for various data types to be instantiated;

a process for detecting the multiphase data type definition declared in the preprocessed source program;

a process for registering, if a name of the detected data type definition is not registered, the name of the detected data type definition into the multiphase data type definition table;

a process for deleting, if the name of the detected data type definition is registered, the multiphase data type definition in the source program;

a process for setting, if the multiphase data type definition is described in a body of any of the source programs to be linked into the one object program, the use flag to a use status;

deleting the multiphase data type definition of which the use flag is set to the use status from all the source program to optimize the source programs;

registering, if instantiation information arranged for each data type of a multiphase data type is not registered, the instantiation information into the data type definition table, the multiphase data type employing a template model for various data types to be instantiated;

generating, if the instantiation information of the data type of which instantiation is requested is not registered, generate no instance of the data type definition in the source program;

~~a process for scanning all the preprocessed source programs to be used as a source for generating the object program;~~

~~a process for deleting, when a multiphase data type definition is already stored or the use flag is not set in a use status in the multiphase data type definition table, the multiphase data type definition from the source programs to optimize the source programs;~~

~~a process for generating instance of the multiphase data type description in the source programs only when the corresponding use flag is not set in a use status in the multiphase data type definition table;~~ and

a process for compiling the optimized source programs.